(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 168 162 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
02.01.2002 Bulletin 2002/01

(51) Int Cl.[7]: **G06F 9/44**

(21) Application number: **01304358.3**

(22) Date of filing: **16.05.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR.
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **30.06.2000 US 607369**

(71) Applicant: **Nokia Corporation**
02150 Espoo (FI)

(72) Inventor: **Paajanen, Samu**
00150 Helsinki (FI)

(74) Representative: **Read, Matthew Charles et al**
Venner Shipley & Co. 20 Little Britain
London EC1A 7DH (GB)

(54) Tag-based user interface

(57) The invention provides a method for converting one machine readable language into another by applying instructions to style templates. The invention provides a high level construct created from a set of lower lever tags. A man machine interface (MMI) may be created by using said constructs.

EP 1 168 162 A2

Printed by Jouve, 75001 PARIS (FR)

Description

FIELD OF THE INVENTION

[0001] This invention relates to user interfaces, and in particular an user interface wherein the user views are generated dynamically.

BACKGROUND OF THE INVENTION

[0002] World Wide Web (WWW) applications have been implemented so that each user view, or screen, is one entity. An entry is usually a static file created in a mark-up language such as HyperText Mark-up Language (HTML), eXtensible Mark-up Language (XML) and the like or mark-up language with some dynamically created parts such as a JSP or ASP file. Dynamic HTML (DHTML) is made up of HTML, JavaScript, and Cascading Style Sheets (CSS) and allows HTML page to be dynamically changed.

[0003] This implementation works well with information on a page. However, if content on the page changes dynamically, or the developer would like to place interactivity such as buttons and the like, the implementation fails. Therefore it is difficult to create a user interface views in mark-up language.

[0004] The coding the parts of a user interface can be time consuming, error prone, and unmanageable. What is needed are higher level constructs which will allow for fast and easier programming.

[0005] Thus, the need arises, for a method and constructs which will allow an dynamic web content to be developed in less time and with less trouble.

SUMMARY OF THE INVENTION

[0006] In accordance with the present invention, there is provided a method for generating display instructions by converting a templates containing style information into a final form which converts data created in one language into a second language.

[0007] Also provided is a high level construct created by a set of lower level mark-up language tags.

[0008] Additional features and advantages of the invention will be readily apparent from the specification and from the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009]

Fig. 1 is a block diagram representation of a communication network having mobile terminals that are capable of communicating with a mobile display appliance (MDA) system having a MDA server with related services.

Fig. 2 is a more detailed block diagram representation of the MDA server of the MDA system of the network of Fig.1.

Fig. 3 is a more detailed block diagram of a mobile terminal that operates within the MDA system of Fig. 1

Fig. 4 is an illustration of the component model of the present invention.

Fig. 5 is a flowchart showing the page generation process in accordance with an embodiment of the current invention.

Fig. 6 shows an example of localized component specifications in accordance with an embodiment of the current invention.

DETAILED DESCRIPTION.

[0010] Referring now to Fig. 1, a network 10 includes a terminal 20 coupled to an access point 22. The access point 22 is coupled to an Internet Service Provider (ISP) 24, which is coupled to an Internet 26. Accordingly, the access point 22 has an internet address relative to the internet address of the ISP 24. Additionally, the ISP 24 is coupled to a Mobile Display Appliance (MDA) server 28 that provides the user, through the terminal 20, with specific services and features, which will be discussed in detail herein.

[0011] The terminal 20 may be a hand held mobile radio telecommunications device that includes a virtual keyboard,

a two-fingered navigational tool and a two-fingered pressure sensitive special click-drag-drop feature. The MDA server 28 provides services such as email, calendar, notes, ability to shop on line and necessary authentication, as well as third party services and information.

[0012] Terminals 20, 20a, and 20b are coupled to the access point 22 via wireless connections 30, 30a, and 30b, respectively, and, hence, the user has portable or mobile access to the Internet 26 and the services provided by the MDA server 28. Additionally, a personal computer (PC) terminal 21 is coupled to the access point 22 via a land line 31. The terminal 21 can be used to access the MDA server 28 using special authentication by any user authorized to access the information and services provided by the MDA server 28. However, the authentication for the user using the terminal 21, which is discussed herein, is slightly different from the authentication procedure for the terminals 20, 20a, 20b, and 20c. More specifically, the terminal 20 is coupled to the access point 22 using a Wireless Local-Area-Network Gateway (WLAN GW) that is installed at a specific location, such as the user's premises or location. In the preferred embodiment, the WLAN GW interface uses Ethernet 802.11 transfer protocol. However, other wireless interface protocols, such as GPRS of Global System for Mobile Communications (GSM+), Universal Mobile Telecommunication Systems (UMTS), or other LAN, may be used without limiting the scope of the present invention as set forth in the claims hereinafter. If the terminal 20 is powered on and within range of the access point 22, then Ethernet protocol is used as a transfer protocol in order to establish and maintain a communication link.

[0013] Although the preferred embodiment shows the terminal 20 coupled to the MDA server 28 through the ISP 24, the scope of the present invention, as set forth in the claims, is not limited thereby. For example, the terminal 20 may be coupled directly to the MDA server 28 through the access point 22. Regardless of how the terminal 20 is coupled to the MDA server 28, once the terminal 20 is authenticated, as will be discussed herein, it can function as an Internet browser to access the Internet 26 with the additional ability to retrieve services and information from the MDA server 28. Furthermore, in the embodiment set forth herein, the ISP 24 is separate from and not acting as the MDA server 28 and vice versa, even though this is possible to combine them into one unit.

[0014] It will be apparent to those skilled in the art that even though the preferred embodiment shows that the access point 22 coupled to the ISP 24 through a land line 32, the scope of the present invention as set forth in the claims is not limited thereby. For example, the access point 22 can be wirelessly coupled to the ISP 24. Thus, in the preferred embodiment, the terminal 20 accesses the ISP 24 through the access point 22 and, thus, the user can access, navigate through, and retrieve information from the Internet 26 using the terminal 20.

[0015] In order for a terminal, such as terminal 20, to have access to the services of the MDA server 28, the MDA server 28 must authenticate and authorize the terminal's access. Although only the authentication and authorization procedure relating to the terminal 20 are discussed in detail, the teachings set forth herein are also applicable to other terminals. Upon proper authentication of the terminal 20 the user can access the services of the MDA server 28 at the authorized level of authentication.

[0016] Generally stated, if the terminal 20 is powered on and authenticated by the MDA server 28, then information or services from the MDA server 28 are downloaded to the terminal 20. The MDA server 28 downloads information, such as profile setting for the family. One profile setting that can be downloaded is language preferences for a family communication session. Other information or services may include configuration data, driver or application related software or portions thereof, configurable parameters, partial sections of system software, or configurable parameters depending on the level of authentication that has occurred with respect to the user. Additionally, the terminal can have access through proper authentication and service purchases to third party publications available from a vendor 33, such as news related information found in magazine publications or the daily newspaper. It will be apparent to those skilled in the art that the information may be purchased by the user and then transmitted by the vendor 33 upon request of the user's family at the MDA server 28 and then to all terminal within the terminal 20; alternatively, the information could be purchased by an operator/owner of the services provided by the MDA terminal 28 and then resold to each family as requested. Thus, a family profile could may also include access to the information services of the vendor 33 that can be made available to the family or just the user, depending on the authentication.

[0017] There are two levels of authentication that provide access to the services and information of the MDA server 28: the family level and the private level. The family level is a level of authentication that occurs based on the identity of the terminal 20 in order to initiate a family session. In order to create a family, at least one terminal is needed, but typically there are several terminals that make up a family, such as terminals 20a, 20b, and 20c, and each terminal has a unique identity that allows that terminal access to a family session at the family level. Furthermore, each family includes a specific family profile and this family profile is downloaded during a family session from the MDA server 28. Thus, anyone having access to the terminal 20 would have access to the family level information and services, such as calendar, e-mail, bookmarks, cookies, and e-publication, all of which are set up for the family. As will be discussed herein, these same services may be available to the user at the private level, but the content of the information would vary. The MDA server 28 includes storage capacity for storing data related to the family in a family specific storage unit that can be accessed and used by all terminals within the family once the terminal has been authenticated and the family session initiated.

[0018] In the preferred embodiment, the family level authentication is based on the identity of the hardware of the terminal 20 and the authentication occurs automatically to initiate the family session once the terminal 20 is powered on. Even though the authentication at the family level occurs automatically, the scope of the invention as set forth in the claims is not limited thereby. For example, the terminal 20 could request input from the user in order to initiate the family level authentication process. Once the terminal 20 is authorized to access the services, then each user of the terminal 20 is able to access information and services that are available to all users in the family as well as initiate a private communication session to access private information and services available only to that user, provided the user has a profile associated with family associated with the terminal that has established the family session that is in progress.

[0019] Unlike the family session at the family level, a private session at the private level is a level of authentication that requires input from the user to allow the user of a terminal, such as the terminal 20 or the terminal 20c, access to information intended only for that user. For example, the user could use any terminal that is within the user's family and initiate a private session to access information and services specific to that user. The authentication can be done using anything that is unique and only known by that user, such as a password. Thus, the user can initiate a private session regardless of which terminal is being used. When the user activates a private session then configuration parameters, which are specific to the user, are downloaded to the terminal 20. Although in the preferred embodiment a user must be have a profile associated with the same family that the terminal's profile is associates, the scope and sprite of the present invention is not limited thereby. For example, the network 10 could be set up to allow a user access from any terminal regardless of the association between the user, the terminal 20, and the family as long as the user can be authenticated by the MDA server 28. This is similar to the way a user would gain access to the MDA server 28 from the terminal 21.

[0020] As indicated above, anyone having access to the terminal 20 would have access to the family level information and services, because authentication is based on the terminal 20 and occurs automatically and the family session is always active when the terminal 20 is powered on. Even though any user of the terminal 20 can have access to information and services at the family level, only a designated user can change the family or take actions on behalf of the family. In the preferred embodiment, one or two users within the family are typically designated to have administrative rights for the family. The user/users with administrative rights are called a family administrator. The family administrator has the right to alter family profiles and the information related to the family administrator is stored in the MDA server 28 and administration access can be authenticated by a password. The family administrator, once authenticated, can alter the family profile settings, add or delete terminal profiles form the family profile, and add or delete user profiles form the family profile.

[0021] One family setting that the family administrator can select is the language setting for the family sessions. However, each user can select his or her own language preference for the private sessions. For example, in a multilingual family the family language can be one specific language, while the language for each user may be different during the private session for that user. Then depending on the session type, which is either family or private, the terminal 20 will show text in the selected language, which is established in the family or private profile, respectively.

[0022] Additionally, the family administrator can have access to purchasing services that may require the ordering party to be of legal age for the purpose of ordering or purchasing additional services, such as news or publication services. Thus, while all users of the terminal would have access to family level services, such as access to the internet, they would not be able to make administrative decision, unless they were authenticated as the family administrator. Accordingly, the family is protected from unauthorized or unwanted alteration of family profile as well as financial commitments from occurring at the family level from the terminal 20, especially given that the identity of the user of the terminal 20 is not unknown at the family level yet that user has access to the MDA server 28 as well as the internet 26 through the terminal 20 coupled to the ISP 24.

[0023] Continuing with Fig. 1, in addition to the ISP 24, the access point 22 is also coupled to a global unit or business owner 34. As indicated, the access point 22 may be coupled directly to the business owner 34 through a link 35a. Alternatively, the access point 22 may be coupled indirectly to business owner 34 through a land line 32, the ISP 24, and a link 35b.

[0024] The business owner 34 includes a global address server 36, a global upgrade server 38, and a firewall unit 40. It will be apparent to those skilled in the art that the firewall unit 40 functions to provide secured access to the global address server 36 and the global upgrade server 38.

[0025] In the preferred embodiment, the internet address of the business unit 34 with the global address sever 36 is permanently contained in the memory of the terminal 20. Even though reference is made hereinafter only to the internet address of the global address server 36 without specific reference to the internet address of the business owner 34, it will be apparent to those skilled in the art that the internet addresses for the two may be the same or could be slightly different depending on configuration parameters. The global address server 36 is a place from which all the terminals, such as terminals 20, 20a, 20b, and 20c, can fetch the internet address of their respective MDA server. The advantage of having the terminal 20 store the internet address of the global address server 36 is that if the terminal

20 was relocated near another access point, then the terminal 20 can still obtain the internet address location of the MDA server 28 simply by knowing the internet address of the global address server 36. However, the scope of the invention as set forth in the claims is not limited thereby. For example, the internet address of the MDA server 28 could be stored on the terminal 20 and the memory of the terminal 20 could be updated as needed.

[0026] An advantage to storing the internet address of the global address server 36 on the terminal 20 is that the association between terminal and MDA server as well as changes in the internet address of MDA servers can be easily and efficiently updated without having to update the memory of each terminal. The global update server 38 updates the global address sever 36 each time there is a change in the association between terminal and MDA server, when there are new terminals to associate with an MDA server, or if the internet address of a particular MDA server is changed.

[0027] With the internet address of the global address server 36 stored in the memory of the terminal 20, the terminal 20 is able to request and retrieve the internet address of the MDA server 28 from the global address sever 36. The global address server 36 stores information about the location of the MDA server 28 and all other MDA servers in the network and the corresponding relation between each terminal and its MDA server. Thus, the terminal 20 is always able to obtain the address of the MDA server 28, which is the MDA server designed to serve the terminal 20. For example, the terminal 20c coupled through an access point 42 to an ISP 44 can retrieve the internet address of the MDA server 28 from the global address server 36, provided that the MDA server 28 is the MDA server designated to serve the terminal 20c and that the terminal 20c is authenticated by the MDA server 28 as an authorized user of the services.

[0028] Referring now to Fig 2, the MDA server 28 includes a support server 46, an application server 48, a network application server 50, and a directory server 52. It will be apparent to those skilled in the art that the referenced connections do not depict the physical connections between the logical elements; the emphasis is merely on the logical connections. The support server 46 provides services oriented towards enabling and supporting the services provided to the terminal 20. The support server 46 includes an upgrade service unit 54, a login services unit 56, a profile services unit 58, an advertisement services unit 60, an administrative services unit 62, a defined services unit 64, and a directory client unit 66.

[0029] The upgrade services unit 54 is a specific means for controlled software upgrade of the software for the support server 48. Updates are transmitted from the global upgrade server 38 to the upgrade service unit 54. The login services unit 56 provides the means for authentication of the user and the terminal 20 that is being used to access the services based on information provided by the client unit 66. Additionally, the login services unit 56 is also responsible for log-off activities, such as private session termination. The profile services unit 58 provides a means for modifying a user's profile information, e.g. family and private information and preferences. The administration services unit 62 provides a means for administration of the support server 46 and the application server 48. The advertisement services unit 60 provides a means for the MDA server 28 to tailor advertisements to the user and the terminal 20 according to the user's profile information. The defined services unit 64 is a classification of 'other services' containing items like bookmark management services, help services, log services, name management services, and general management services. The directory client unit 66 is coupled to the directory server 52 to provide client verification

[0030] Referring now to Fig. 3, the terminal 20 includes a display 70, a user interface (UI) framework 72, a browser 74, a driver 76, and hardware 78. The user interface—which may also be referred to as a Man-Machine Interface (MMI)--can be logically divided into components (or constructs), such as buttons, groups of buttons, folders, folder items and the like. Each component may be constructed of a set of sub-components.

[0031] In different description languages, such as HTML and the like, a user interface may be constructed of a set of tags. These tags may be grouped and the new group be represented as a new tag. This tag may be referred to as a component. Thus, a set of new tags may be in turn group to form a novel higher level component.

[0032] In web applications, the components from which the user interface is constructed may unfolded and stored in the server before the application is started.

[0033] Creating an user interface by using the novel higher level components of the present invention requires less code thereby making development of the user interface easier, faster and less error-prone. These new higher level components also make the resulting code is also more manageable by reducing the amount of lines needed to describe the user interface.

[0034] Local specific tags and text may be hidden in the components making localization easy. Additionally, the local settings may be changed in run-time. Therefore, web pages may be local in run-time; thereby, the language and other local setting can be easily changed.

[0035] Referring again to Fig. 3, the driver 76 resides in the memory of the hardware 78 along with other data, such as the internet address of the global address server 36 and software, such as the browser 74. As the terminal 20 is turned on, the driver 76 retrieves data relating to the internet address of the global address server 36. In the preferred embodiment, the driver 76 is EPOC6, which is an operating system software that handles hardware related functions in the terminal as well as offer a functioning environment to the application layer programs. Once the terminal 20 is power on, it is coupled to the access point 22 and the ISP 24. Thus, the terminal 20 is able to obtain its own internet

address.

[0036] Using the internet address of the global address server 36, the terminal 20 is coupled to the global address server 36 and sends a request in order to obtain the internet address of the MDA server 28. Once the terminal 20 has the internet address of its MDA server 20, it is then coupled to the MDA server 28. The MDA server 28 authenticates, using the unique identity of the hardware 78 of the terminal 20, that the terminal 20 has family level access privileges. Accordingly, the terminal 20 is authenticated and logged onto the MDA server 28 to begin a family session at a family level. Thus, the user can now access services or retrieve information from the MDA server 28 or the Internet 26. In order for the user to initiate a private session and retrieve private information, the user must use the terminal 20 and provide further authentication to the MDA server 28 to gain access at the private level. It will be apparent to those skilled in the art that at either the family level or the private level, the user is able to the retrieve information related to the family of users as well as browse the Internet 26 to retrieve information.

[0037] The browser 74 is a typical browser and includes such features as HyperText Transfer Protocol (HTTP), JAVA script, and cascade style sheet capability. As with typical Personal Computers (PCs), the browser 74 helps the user navigate through and retrieve information from the Internet once the user is connected to the ISP 24 through the terminal 20. The user utilizes the terminal 20 to connect to both the ISP 24 and the MDA server 28 using authentication protocol as discussed in detail herein. The terminal 20 is the primary means of access by the user to the MDA server 28 and the related services and applications. However, the user can also access the ISP 24 and the MDA server 28 using the terminal 21 or non-mobile terminal using appropriate family level authentication initiated manually.

[0038] In order to retrieve information or request services from the MDA server 28 or the Internet 26, the user provides input through the UI framework 72. The user can provide input using a virtual keyboard displayed on the display 70. Even though the virtual keyboard is used as the user retrieves information from the Internet 26, such as a web page, the user can receive the information at the display 70 of the terminal 20 in a full screen format. Full screen format is available because the UI framework 72 disappears when the user types a Universal Resource Locator (URL) or follows a hyperlink while navigating the Internet 26. In order to return to the UI framework 72, the user presses a button 80 and the virtual keyboard as well as the header and footer related to the services are presented again. Additionally, once the user presses the button 80 the web page, which was a full screen displayed prior to pressing the button 80, is reduced to a thumbnail view and positioned in the display 70, such as in the bottom left corner of the footer. Consequently, the user has a shortcut to quickly access the web page that was previously visited or to save that web page as a bookmark.

[0039] In a preferred embodiment of the present invention, content and layout are separated. The content is XML or other similar mark-up language, which is usually dynamically generated by different services. Screen layouts are defined in eXtensible Style Language (XSL) or other similar style language. Additionally, XSL layouts are composed of components, which are sets of lower-level tags.

[0040] User views are created dynamically by putting together the XML and the XSL. The XSL data is also constructed dynamically. Higher level component tags are unfolded in runtime. The unfolded XSL files are cached in memory in Java Virtual Machine's internal object form for fast response times.

[0041] Fig. 4 is an illustration of the component model in accordance with an embodiment of the present invention. View 100 may contain a plurality (1 to n : with n being an integer) of possible components. The UI component 110 is used in at least one view, but it can be used in any number of views (1 to n). Furthermore, the same component may be inside many components or none (thus, 0 to n).

[0042] There are three types of components which may be used in XSL templates. The following tag names are used for these types:

COMPONENTS_COMP
COMPONENTS_LOCAL
COMPONENTS_SCRIPT

[0043] COMPONENTS_COMP may contain any set of tags,
COMPONENTS_LOCAL may be used for localization. COMPONENTS_LOCAL may contain basically any tags which need to be localized, but mostly COMPONENTS_LOCAL contains text strings. COMPONENTS_SCRIPT is a fragment of script language, e.g. JavaScript and the like.

[0044] This invention has been described relative to specific embodiments. The MDA environment in which the preferred embodiment operates is used as an exemplar only. The names of the tags are for use for the MDA terminal. The software examples have been written in Finland and thus the spelling of certain words may be different. For example, LOCALE is spelled with an E instead of LOCAL. Modifications that become apparent to persons of ordinary skill in the art only after reading this document are deemed within the spirit and scope of the invention.

[0045] In the preferred embodiment of the present invention, the following tag names are used for these types:

MDA_COMPONENT
MDA_LOCALE
MDA_SCRIPT

[0046] As mention above but repeated with the preferred embodiment names. MDA_COMPONENT may contain any set of tags, MDA_LOCALE may be used for localization. MDA_LOCALE may contain basically any tags which need to be localized, but mostly MDA_LOCALE contains text strings. MDA_SCRIPT is a fragment of script language, e.g. JavaScript and the like.

[0047] Fig. 5 is an illustration of the page generation process. Pages are generated so that the original XSL-template (with components included) 120 goes through three conversions 130, 140, and 150. In the first conversion 130, all components of type MDA_COMPONENT are unfolded using instructions which are given by MDA_COMPONENT_XSL 125. First conversion 130 results in converted XSL-template-1 133. The second conversion 140 unfolds all MDA_LOCALE components using instruction provided by MDA_LOCALE 135. Second conversion 140 results in converted XSL-template-2 143. A third conversion 150 unfolds MDA_SCRIPT components using instructions provided by MDA_SCRIPT_XSL 145. Third conversion 150 results in final XSL template 153 . The above described process is referred to as XSL generation and results, as stated above, final XSL template 153, which is cached into memory of terminal 20.

[0048] Final template 153 may then be combined with the XML data 155 returned by the services which are shown in figs. 1 and 2 and described in co-pending United States Patent Application Serial Number_____ filed on _____, entitled NETWORK WITH MOBILE TERMINALS HAVING WIRELESS ACCESS TO THE INTERNET AND METHOD FOR DOING SAME and herein incorporated by reference. A last conversion 160 takes the XML data 155 which may be instructions and combines it with the final template 153 which results in instructions to create a final user view in the case of our example—HTML component.

[0049] Components of type MDA_COMPONENT may contain other components. However, there must not be any unending recursion. Components of type MDA_COMPONENT are processed recursively as is shown in fig. 4. The instructions provided by MDA_COMPONENT_XSL are applied to all components of type MDA_COMPONENT until all such components are unfolded. There is no need to process the components of type MDA_LOCALE or MDA_SCRIPT recursively because they can never contain other component tags.

[0050] In run time environment, there is only one file for the definitions of components of type MDA_COMPONENT. For the components of type MDA_LOCAL and MDA_SCRIPT there is one file of each supported language. Fig. 6 shows the localized component specifications.

[0051] Referring to Fig. 6, There is one COMPONENT type file MDA_COMPONENT_XSL. However, there may be one LOCALE type 210 for each language. In this example, MDA_LOCALE_XSL_EN_en for English 213, MDA_LOCALE_XSL_US_en for Standard American English 215, and MDA_LOCALE_XSL_FI_en for FINNISH 217. There can also be multiple SCRIPT type 220 files. For example, MDA_ SCRIPT_XSL_EN_en for English 223, MDA_ SCRIPT_XSL_US_en for Standard American English 225, and MDA_ SCRIPT_XSL_FI_en for Finnish 227.

[0052] The following are example of code to build views using XSL and XML, other style and mark-up languages and modifications may be used without departing from the spirit and scope of the invention.

A first example:

[0053] In this first example an HTML-based UI-layout (file UI_EXAMPLE) contains one component of type MDA_COMPONENT, and two components called MDA_LOCALE. The definitions of the components are given in XSL-files called COMPONENTS_COMP and COMPONENTS_LOCALE.

[0054] When component definitions given in COMPONENTS_COMP are applied to file UI_EXAMPLE, the components of type MDA_COMPONENT are replaced with the sets of tags specified in file COMPONENTS_COMP. In this first example, there is just one component called "form1". The resulting file is called CONVERTED_UI_EXAMPLE.

[0055] CONVERTED_UI_EXAMPLE contains components of type MDA_LOCALE. When COMPONENTS_LOCALE is applied to CONVERTED_UI_EXAMPLE, the components of type MDA_LOCALE are replaced with texts and tags given in file COMPONENTS_LOCALE. The resulting file is called FINAL_RESULT_UI_EXAMPLE.The files are shown below:

UI_EXAMPLE

```
<HTML>
 <HEAD>
  <TITLE><MDA_LOCALE TEXT="FORM_TEST"/></TITLE>
 </HEAD>

 <BODY BGCOLOR="#FFFFFF">
  <H2><MDA_LOCALE TEXT="FORM_TEST"/></H2>
  <MDA_COMPONENT TYPE="form1" NAME="my_form"/>
 </BODY>
</HTML>
```

COMPONENTS_COMP

```
<?xml version="1.0"?>
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:output encoding="ISO-8859-1"/>

 <xsl:template match="MDA_COMPONENT">
  <xsl:choose>
   <xsl:when test="@TYPE='form1'">

    <FORM>
     <xsl:attribute name="name">
      <xsl:value-of select="@NAME"/>
     </xsl:attribute>
     <TABLE>
      <TR>
       <TD><MDA_LOCALE TEXT="FIRST_NAME"/></TD>
       <TD><INPUT type="text" name="date_field" size="35"/></TD>
      </TR>
      <TR>
```

```
            <TD><MDA_LOCALE TEXT="LAST_NAME"/></TD>
            <TD><INPUT type="text" name="time_field" size="35"/></TD>
          </TR>
          <TR>
            <TD><MDA_LOCALE TEXT="COMMIT_BUTTON"/></TD>
            <TD><MDA_LOCALE TEXT="CANCEL_BUTTON"/></TD>
          </TR>
        </TABLE>
      </FORM>

    </xsl:when>
  </xsl:choose>
 </xsl:template>

 <xsl:template match="*|@*|comment()|processing-instruction()|text()">
  <xsl:copy>
   <xsl:apply-templates select = "*|@*|comment()|processing-instruction()|text()"/>
  </xsl:copy>
 </xsl:template>

</xsl:stylesheet>
```

COMPONENTS_LOCALE

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:output encoding="ISO-8859-1"/>

 <xsl:template match="MDA_LOCALE">
  <xsl:choose>
   <xsl:when test="@TEXT='FIRST_NAME'">First name:</xsl:when>
   <xsl:when test="@TEXT='LAST_NAME'">Last name:</xsl:when>
   <xsl:when test="@TEXT='FORM_TEST'">Form test</xsl:when>
   <xsl:when test="@TEXT='COMMIT_BUTTON'">
    <INPUT type="button" name="submit_button" value="Submit"
onClick="submit()"/>
   </xsl:when>
   <xsl:when test="@TEXT='CANCEL_BUTTON'">
    <INPUT type="button" name="cancel_button" value="Cancel"
onClick="cancel()"/>
   </xsl:when>
  </xsl:choose>
 </xsl:template>
```

```
<xsl:template match="*|@*|comment()|processing-instruction()|text()">
  <xsl:copy>
    <xsl:apply-templates select = "*|@*|comment()|processing-instruction()|text()"/>
  </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

CONVERTED_UI_EXAMPLE

```
<HTML>
 <HEAD>
  <TITLE><MDA_LOCALE TEXT="FORM_TEST"></MDA_LOCALE></TITLE>
 </HEAD>
 <BODY BGCOLOR="#FFFFFF">
  <H2><MDA_LOCALE TEXT="FORM_TEST"></MDA_LOCALE></H2>
  <FORM name="my_form">
   <TABLE>
    <TR>
     <TD><MDA_LOCALE TEXT="FIRST_NAME"></MDA_LOCALE></TD>
     <TD><INPUT size="35" name="date_field" type="text"/></TD>
    </TR>
    <TR>
     <TD><MDA_LOCALE TEXT="LAST_NAME"></MDA_LOCALE></TD>
     <TD><INPUT size="35" name="time_field" type="text"/></TD>
    </TR>
    <TR>
     <TD><MDA_LOCALE
TEXT="COMMIT_BUTTON"></MDA_LOCALE></TD>
     <TD><MDA_LOCALE
TEXT="CANCEL_BUTTON"></MDA_LOCALE></TD>
    </TR>
   </TABLE>
  </FORM>
 </BODY>
</HTML>
```

FINAL_RESULT_UI_EXAMPLE

```
<HTML>
 <HEAD>
  <TITLE>Form test</TITLE>
 </HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
 <H2>Form test</H2>
 <FORM name="my_form">
  <TABLE>
   <TR>
    <TD>First name:</TD>
    <TD><INPUT size="35" name="date_field" type="text"></TD>
   </TR>
   <TR>
    <TD>Last name:</TD>
    <TD><INPUT size="35" name="time_field" type="text"></TD>
   </TR>
   <TR>
    <TD><INPUT onClick="submit()" value="Submit" name="submit_button"
          type="button"></TD>
    <TD><INPUT onClick="cancel()" value="Cancel" name="cancel_button"
          type="button"></TD>
   </TR>
  </TABLE>
 </FORM>
</BODY>
</HTML>
```

[0056] As with any style sheet or mark-up language, all XSL-files should tell the XML version and type of desired output. In the example of an embodiment of the present invention, this may be accomplished by using the following two lines:

```
<?xml version='1.0'?>

<?xml-stylesheet type="text/xml"?>
```

[0057] The XSL definition has a root element *xsl:stylesheet* with attributes *version* and *xmlns:xsl.* An example of declaring XML namespace for an embodiment of the current invention is as follows:

```
<xsl:stylesheet version="1.0" xmlns:xsl=http://www.w3.org/1999/XSL/Transform>
```

[0058] An advantage of the present application is that the language may be changed locally at run-time. Therefore, a second example is provided which shows the use of the present invention using Finnish language as the example text.

[0059] In order to support western and scandinavian character sets, the output coding must be set to ISO-8859-1, e.g.:

```
<xsl:output encoding="ISO-8859-1"/>
```

[0060] Thus, XSL-files may be of the form:

```
<?xml version="1.0?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmins:xsl="http//www.w3.org/1999/xsl/Transform">
  <xsl:ouput encoding="ISO-8859-1"/>

    <!—all XSL-definitions here-->

</xsl:stylesheet>
```

The XML files are of the form:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>

  <!—all XML-data here-->
```

[0061] The following is an example of implementation of XSL templates. In an embodiment of the present invention, Finnish component versions are used. Thus, the component specifications are given in files:

MDA_COMPONENT_XSL_FI_fi,

MDA_LOCALE_XSL_FI_fi, and

MDA_SCRIPT_XSL_FI_fi.

[0062] These files are all actually XSL files, which tell the processor to copy all the content from the input stream to the output stream, and replace some tags with something else. These are called component specification files.

[0063] For example, if there is to be no change to any component tags, all the component specification files may look like this:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/199/XSL/Transform">
   <xsl:output encoding="ISO-8859-1"/>
   <xsl:template match=*|@*|comment()|processing-instruction()|text()>
      <xsl:copy>
         <xsl:apply-templates select = *|@*|comment()|processing-
instruction()|text()/>
      </xsl:copy>
   </xsl:template>

</xsl:stylesheet>
```

Example Listing 1: Empty component specification file

[0064] This component specification file simply orders the processor to copy all the input to the output as it is.

[0065] Let us use the following XSL template as an example in the following chapters.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/199/XSL/Transform">
   <xsl:output encoding="ISO-8859-1"/>
    <xsl:template match="CLOCK">
      <HTML>
        <HEAD>
```

```
<TITLE><MDA_LOCALE TEST="KELLO"/></TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    <MDA_SCRIPT TYPE="clockscript" REFERS="clock1"/>
  </SCRIPT>
 <HEAD>

 <BODY BGCOLOR="#FFFFFF">
  <H2><MDA_LOCALE TEXT="KELLO"/></H2>
    <MDA_COMPONENT TYPE="clock" NAME="clock1"/>
    <xsl:apply-templates/>
 </BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

Example Listing 2: XSL template 120.

[0066] The example file of our preferred embodiment contains four component tags:

```
<MDA_LOCALE TEXT="KELLO"/>,
<MDA_SCRIPT TYPE="clockscript" REFERS="clock1"/>,
<MDA_LOCALE TEXT="KELLO"/>, and
<MDA_COMPONENT TYPE="clock" NAME="clock1"/>,
```

XSL COMPONENTS

[0067] Referring again to Fig. 5, in first conversion 130, if a match to the parameter "type" is found from the component specification file MDA_COMPONENT_XSL 125, then all component tags of type MDA_COMPONENT are replaced with a specified set of tags. If no match to the parameter "type" is found from the component specification file MDA_COMPONENT_XSL 125, then the MDA_COMPONENT tag is discarded.

[0068] An example component specification file which processes MDA_COMPONENT is shown below.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/199/XSL/Transform">
  <xsl:output encoding="ISO-8859-1"/>

    <xsl:template match="MDA_COMPONENT">
      <xsl:choose>
        <xsl:when test="@TYPE='clock'">

          <FORM>
            <xsl:attribute name="name".
              <xsl:value-of select="@NAME"/>
            </xsl:attribute>
            <TABLE>
              <TR BGCOLOR="DDEEAA">
                <TD>MDA_LOCALE TEXT="TANAAN_ON"/>,/TD>
                <TD><INPUT type="text" name="date_field" size = "35" value = " "
                    /></TD>
              </TR>
              <TR>
                <TD>MDA_LOCALE TEXT="KELLO_ON_NYT":<TD>
                <TD><INPUT type="button" name=time_field" size = "35" value = " "
                    /></TD>
              </TR>
              <TR>
                <TD><MDA_LOCALE TEXT="FORM_FIELD1"/></TD>
                <TD>< MDA_LOCALE TEXT="FORM_FIELD2"/></TD>

              </TR>
            </TABLE>
          </FORM>
        </xsl:when>
```

```
    </xsl:choose>
  </xsl:template>

  <xsl:template match=*|@|comment()|processing-instruction()|text()>
    <xsl:copy>
      <xsl:apply-templates select = *|@*|comment()|processing-
                                    instruction()|text()/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Listing 3: Component specification file for MDA_COMPONENT tags MDA_COMPONENT_XSL 125.

[0069] The component specification file MDA_COMPONENT_XSL 125 only replaces the MDA_COMPONENT tags having "clock" as a value of their "type" parameters. Such tags are replaced with HTML Form specification. In the example described in accordance to an embodiment of the current invention, the tag <MDA_COMPONENT TYPE="clock" NAME="clock1"/> is replaced with a form.

[0070] After first conversion 130, the XSL template 120 (example listing 2), is now converted XSL-template 133 and looks like this:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/199/XSL/Transform">
  <xsl:output encoding="ISO-8859-1"/>

    <xsl:template match="CLOCK">

    <HTML>
      <HEAD>
        <TITLE><MDA_LOCALE TEXT="KELLO"/></TITLE>
        <SCRIPT LANGUAGE="JavaScript">
        <MDA_SCRIPT TYPE="clockscript" REFERS="clock1"/>
```

```
    <SCRIPT>
  </HEAD>

  <BODY BGCOLOR="#FFFFFF">
  <H2>
  <MDA_LOCALE TEXT="KELLO"/></H2>
  <FORM name="clock1">
      <TABLE>
        <TR BGCOLOR="DDEEAA">
          <TD>MDA_LOCALE TEXT="TANAAN_ON"/>,/TD>
          <TD><INPUT type="text" name="date_field" size = "35" value = " "
              /></TD>
        </TR>
        <TR>
          <TD>MDA_LOCALE TEXT="KELLO_ON_NYT":<TD>
          <TD><INPUT type="button" name=time_field" size = "35" value = " "
              /></TD>
        </TR>
        <TR>
          <TD><MDA_LOCALE TEXT="FORM_FIELD1"/></TD>
          <TD>< MDA_LOCALE TEXT="FORM_FIELD2"/></TD>

        </TR>
      </TABLE>
    </FORM>
    </xsl:apply-templates>
  </BODY>
 </HTML>
</xsl:template>

</xsl:stylesheet>
```

Listing 4 : XSL template 133 after the first conversion 130.

XSL LOCALES

[0071] Referring once again to Fig. 5, in second conversion 140, all the component tags of type MDA_LOCALE are replaced with given text. If a match to the parameter "text" is found from the component specification file

MDA_LOCALE_XSL 135, then all component tags of type MDA_LOCALE are replaced with a specified set of tags. If no match to the parameter "text" is found from the component specification file MDA_ LOCALE_XSL 135, then the MDA_LOCALE tag is discarded.

[0072] Component specification file MDA_LOCALE_XSL 135 as described below replaces the MDA_LOCALE tags with localized texts.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/199/XSL/Transform">
  <xsl:output encoding="ISO-8859-1"/>
    <xsl:template match=MDA_LOCALE>
      <xsl:choose>
        <xsl:when test="@TEXT='TANAAN_ON'">T<AA/>n<AA/><AA/>n on</xsl:when>
        <xsl:when test="@TEXT='KELLO_ON_NYT'">Kello on nyt</xsl:when>
        <xsl:when test="@TEXT='KELLO'">Kello</xsl:when>
        <xsl:when test="@TEXT='PALVELULISTA'">Palvelut</xsl:when>
        <xsl:when test="@TEXT='JONKUN_KOTISIVU'">n Kotisivu</xsl:when>
        <xsl:when test="@TEXT='LINKIT'">Parhaat www-sivut</xsl:when>
        <xsl:when test="@TEXT='HAKEMISTON_SISALTO'">Hakemisto</xsl:when>
        <xsl:when test="@TEXT='FORM_FIELD1'"><INPUT type="button"
            name="show_now" value="Aika" onClick="start()"/>
        <xsl:when>
        <xsl:when test="@TEXT='FORM_FIELD2'"><INPUT type="button"
            name="clear_now" value="Tyhjennå" onClick="stop()"/>
        <xsl:when>
      </xsl:choose>
    </xsl:template>

      <xsl:template match=*|@|comment()|processing-instruction()|text()>
        <xsl:copy>
          <xsl:apply-templates select = *|@*|comment()|processing-
                    instruction()|text()/>
        </xsl:copy>
      </xsl:template>

  </xsl:stylesheet>
```

Listing 5: The component specification file for MDA_LOCALE tags 135.

[0073] In the example of an implementation of an embodiment of the current invention, on the tag <MDA_LOCALE TEXT="TANAAN ON"/> is replaced with Finnish text "Tånåån on", the tag <MDA_LOCALE TEXT="KELLO_ON_NYT"/ > is replaced with Finnish text "Kello on nyt", and the tag <MDA_LOCALE TEXT="KELLO"/> is replaced with Finnish text "Kello" and so forth.

[0074] In second conversion 140, the component specification file 125 (example listing 5) is applied to XSL template 133 (example listing 4) resulting in converted XSL template2 143.

```
<?xml version="1.0" encoding="ISO-8859-1>
<?xml-stylesheet type="text/xml"?>
<xsl:styleshéet xmlns:xsl="http://www.w3.org/199/XSL/Transform" version="1.0">
  <xsl:output encoding="ISO-8859-1"/>

    <xsl:template match="CLOCK">

    <HTML>
      <HEAD>
        <TITLE>Kello</TITLE>
        <SCRIPT LANGUAGE="JavaScript">
        <MDA_SCRIPT TYPE="clockscript" REFERS="clock1"/>
        <SCRIPT>
      </HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
<H2>Kello</H2>
<FORM name="clock1">
    <TABLE>
      <TR BGCOLOR="DDEEAA">
        <TD> Tånään on:</TD>
        <TD><INPUT value=" " size = "35" name="date_field" type=
          "text"/></TD>
      </TR>
      <TR>
        <TD>Kello on nyt:<TD>
        <TD><INPUT value=" " size="35" name=time_field" type = " " /></TD>
      </TR>
      <TR>
      <TD><INPUT onClick="start() value="Aika" name="show_now"
          type="button"/></TD>
      <TD><INPUT onClick="stop() value=" Tyhjennä " name="clear_now"
          type="button"/></TD>
        TD>
    </TABLE>
  </FORM>
 </xsl:apply-templates>
</BODY>
</HTML>
</xsl:template>

</xsl:stylesheet>
```

Listing 6: XSL template 143 after second conversion 140.

XSL SCRIPTS

[0075] Referring yet again to Fig. 5, in third conversion 150, if a match to the parameter "type" is found from the component specification file MDA_SCRIPT_XSL 145, then all component tags of type MDA_LOCALE are replaced with a specified set of tags. If no match to the parameter "type" is found from the component specification file MDA_SCRIPT_XSL 145, then the MDA_SCRIPT tag is discarded.

[0076] Component specification file MDA_SCRIPT_XSL 145 as described below replaces the MDA_SCRIPT tags with script code.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/199/XSL/Transform">
  <xsl:output encoding="ISO-8859-1"/>
    <xsl:template match=MDA_SCRIPT>
      <xsl:choose>
        <xsl:when test="@TYPE='clockscript'">
        <xsl:text disable-output-escaping="yes">

          &lt;xsltext disable-output-escaping="yes"&gt;
            &amp;lt;!—
              &amp;#10;
              var show_time = false;
              var timeID = null

              &amp;#10;
              function stop(){
                if(show_time){
                  clearTimeout(timerID);
                  document.</xsl:text><xsl:value-of select="@REFERS"/><xsl:text
                disable-output-escaping="yes">.date_field.value = " ";
```

```
      document.</xsl:text><xsl:value-of
    select="@REFERS"/><xsl:text disable-output-
    escaping="yes">.time_filed.value = " ";
      }
      show_time = false;
      }
&amp;#10;

function start(form) {
  var today = new Date();
  var time_display_value = today.getHours();

  if(today.getMinutes() &amp;lt; 10) {
    time_display_value+=:0" + today.getMinutes()
    }
  else{
    time_display_value+=": + today.getMinutes();
    }
  if(today.getSeconds () &amp;lt; 10) {
    time_display_value+=:0" + today.getSeconds()
    }
  else{
    time_display_value+=": + today.getSeconds ();
    }

    date_display_value+=":" + today.getDate() + "." + (today.getMonth() +
  1) + "." + (today.getYear() + 1900);
    document.</xls:text><xsl:value-of select="@REFERS"/><xsl:text
  disable_output-escaping="yes">.time_filed.value = time_display_value;
    document.</xls:text><xsl:value-of select="@REFERS"/><xsl:text
  disable_output-escaping="yes">.date_filed.value = date_display_value;
    timerID = setTimeout("start()",100);
    show_time = true;
    }
```

```
                    —&amp;gt;
                    &lt;/xsl:text&gt;
                    <xsl:text>
                    </xsl:when>

            <xsl:choose>
          </xsl:template>

        <xsl:template match="*|@|comment()|processing-instruction()|text()">
          <xsl:copy>
            <xsl:apply-templates select = "*|@*|comment()|processing-
                                instruction()|text()"/>
          </xsl:copy>
        </xsl:template>

    </xsl:stylesheet>
```

Listing 7: The component specification file 145 for MDA_SCRIPT tags.

[0077] In the example used of an implementation of an embodiment of the current invention, the tag <MDA_SCRIPT TEXT="clockscript" REFERS="clock1/> is replaced with JavaScript code.

[0078] MDA_SCRIPT components must put all the script code inside <xsl:text disable-output-escaping="yes"> begin tag and </text> end tag, as can be seen from the example. The script specification must begin with:

```
<xsl:text disable-output-escaping="yes">
```

followed by:

```
&lt;xsl:text disable-output-escaping="yes"&gt;
&amp;lt;!—
```

[0079] The scrip[t specification must end with:

```
</text>
```

followed by:

```
--&amp;gt;
&lt;/xsl:text&gt;
```

[0080] This inverted texts will result in correct HTML comment tags: "<!—" and "-->".

[0081] In third conversion 150, the component specification file MDA_SCRIPT_XSL 145 (example listing 7) is applied to XSL template 143 (example listing 6). Conversion 150 results in XSL template 153.

```
<?xml version="1.0" encoding="ISO-8859-1"?.?>
<?xml-stylesheet type="text/xml"?>
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/199/XSL/Transform" version="1.0">
  <xsl:output encoding="ISO-8859-1"/>
    <xsl:template match="CLOCK">
    <HTML>

    <HEAD>
      <TITLE>Kello</TITLE>
      <SCRIPT LANGUAGE="JavaScript">


        <xsl:text disable-output-escaping="yes">
          &lt;!—
          &#10;
            var show_time = false;
            var timeID = null

            &#10;
            function stop(){
              if (show_time){
                clearTimeout(timerID);
                document.clock1.date_field.value =" ";
                document.clock1.time_field.value = " ";
```

```
        }
        show_time = false
      }
        &#10;

function start(form) {
  var today = new Date();
  var time_display_value = today.getHours();

  if (today.getMinutes() &lt; 10) {
     time_display_value+=:0" + today.getMinutes()
     }
  else{
     time_display_value+=": + today.getMinutes();
     }
 if (today.getSeconds () &lt; 10) {
     time_display_value+=:0" + today.getSeconds()
     }
  else{
     time_display_value+=": + today.getSeconds ();
     }

     date_display_value+=":" + today.getDate() + "." + (today.getMonth() +
1) + "." + (today.getYear() + 1900);
     document.clock1.time_filed.value = time_display_value;
     document.clock1.date_filed.value = date_display_value;
     timerID = setTimeout("start()",100);
     show_time = true;
     }
     --&gt;
     </xsl:text>

        <SCRIPT>
      </HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
 <H2>Kello</H2>
 <FORM name="clock1">
 <TABLE>
   <TR BGCOLOR="DDEEAA">
     <TD> Tånään on:</TD>
     <TD><INPUT value=" " size = "35" name="date_field" type=
       "text"/></TD>
   </TR>
   <TR>
     <TD>Kello on nyt:</TD>
     <TD><INPUT value=" " size="35" name=time_field" type = "
         "/></TD>
   </TR>
   <TR>
     <TD><INPUT onClick="start() value="Aika"
         name="show_now" type="button"/></TD>
     <TD><INPUT onClick="stop() value=" Tyhjennä "
         name="clear_now" type="button"/></TD>
       </TR>
    </TABLE>
    </FORM>
   </xsl:apply-templates>
  </BODY>
 </HTML>
</xsl:template>
</xsl:stylesheet>
```

Example Listing 8: XSL template 153 after third conversion 150.

[0082] Thus, we have final XSL template 153 (example list 8). When applying it to XML data file 155 (example list 9) we get the HTML page with a simple clock implemented in JavaScript (example listing 10).

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xml"?>
<CLOCK>
</CLOCK>
```

Example Listing 9: XML data 155.

[0083]

```
<HTML>
  <HEAD>
  <TITLE>Kello</TITLE>
  <SCRIPT LANGUAGE="JavaScript">

  <!--

  var show_time = false;
  var timeID = null

  function stop(){
      if (show_time){
        clearTimeout(timerID);
        document.clock1.date_field.value =" ";
        document.clock1.time_field.value = " ";
       }
        show_time = false
      }

  function start(form) {
     var today = new Date();
     var time_display_value = today.getHours();

     if (today.getMinutes() < 10) {
       time_display_value+=:0" + today.getMinutes()
     }
    else{
```

```
    time_display_value+=": + today.getMinutes();
   }
  if (today.getSeconds () < 10) {
    time_display_value+=:0" + today.getSeconds()
  }
 else{
    time_display_value+=": + today.getSeconds ();
   }

   date_display_value+=":" + today.getDate() + "." + (today.getMonth() + 1) + "." +
      (today.getYear() + 1900);
   document.clock1.time_filed.value = time_display_value;
   document.clock1.date_filed.value = date_display_value;
   timerID = setTimeout("start()",100);
   show_time = true;
   }
   -->

 <SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
  <H2>Kello</H2>
     <FORM name="clock1">
        <TABLE>
          <TR BGCOLOR="DDEEAA">
            <TD> Tänään on:</TD>
            <TD><INPUT type= "text" name="date_field" size = "35" value=" "></TD>
          </TR>
          <TR>
            <TD>Kello on nyt:</TD>
            <TD>< INPUT type= "text" name="time_field" size = "35" value=" "></TD>
          </TR>
          <TR>
```

```
<TD><INPUT type="button" name="show_now" value="Aika"
                onClick="start()"></TD>
<TD><INPUT type="button" name="clear_now" value="Tyhjennå "
                onClick="stop()"></TD>
  </TR>
 </TABLE>
</FORM>
</BODY>
</HTML>
```

Example Listing 10 Final HTML component for user view 170.

[0084] Function "start" is called when the user clicks button "aika (Finnish word for "time"). Start function updates the current date and time into the fields of an HTML form. A pointer to the form is sent to the start function as a parameter ("form"). Start function sets the timer so that it is called again after one second. Thus, the time of the form field is updated again and again every second.

[0085] Stop function is called when the user clicks button "Tyhjennå" (Finnish for "clear"). This function clears the form fields and clears the timer.

[0086] This invention has been described relative to specific embodiments and is described with examples of code. The example functions are written according to JavaScript 1.1 standard. However, one skilled in the art after reading the specifications can appreciated that any future version or like language may be substituted without departing from the spirit and scope of the invention. Furthermore, modifications that become apparent to persons of ordinary skill in the art only after reading this document are deemed within the spirit and scope of the invention.

[0087] Applicant's use of the term "plurality" shall be defined as one or more.

Claims

1. A method for generating a template for construction of a document user view comprising the steps of:

   converting a first template containing style information in compliance with a first set of instructions into a second template;
   converting said second template in compliance with a second set of instructions into a third template; and
   converting said third template in compliance with a third set of instructions into a fourth template storing said fourth template into memory.

2. The method of claim 2 further comprising the step of:

   applying said fourth template to data in a first language and converting
   said data in a first language into data of a second language.

3. A computer-readable memory for directing a computer to function in a particular manner when used by the computer, comprising:

   a first portion to direct the computer to convert a first template containing style information in compliance with a first set of instructions into a second template;
   a second portion to direct computer to convert said second template in compliance with a second set of instructions into a third template ;
   a third portion to convert said third template in compliance with a third set of instructions into a fourth template;
   a fourth portion to direct computer to apply said fourth template to data in a first language and convert said data in a first language into data of a second language; and

a fifth portion to direct computer to store said data into memory.

4. A computer data signal embodied in a carrier wave, comprising instructions for:

converting a first template containing style information in compliance with a first set of instructions into a second template;
converting said second template in compliance with a second set of instructions into a third template;
converting said third template in compliance with a third set of instructions into a fourth template; and
storing said fourth template into a memory.

5. The computer data signal of claim 4 further comprising the instruction for:
applying said fourth template to data in a first language and converting said data in a first language into data of a second language.

6. A mobile communications device configured to operate according to a method as claimed in claim 1.

7. A computer program for use with data processing apparatus to perform a method as claimed in claim 1.

8. A computer program according to claim 7 and including code as set out in any one of Examples 1 to 9 of the description.

9. A network configured to operate in accordance with the method claimed in claim 1.

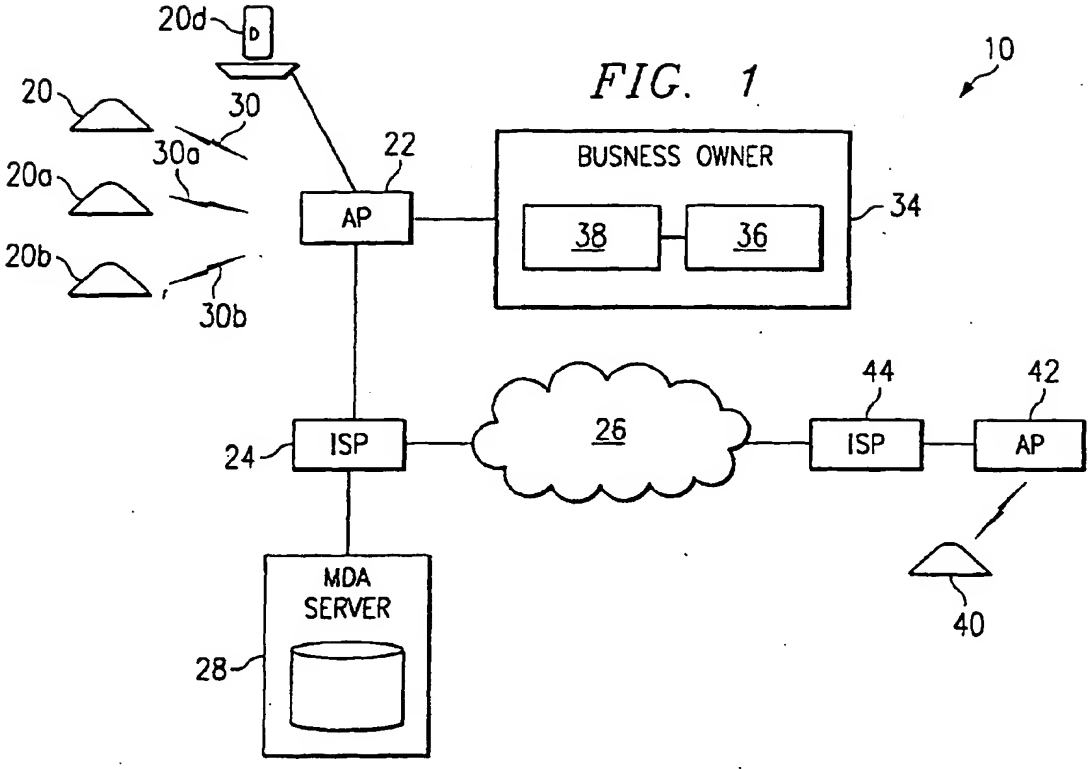


FIG. 1

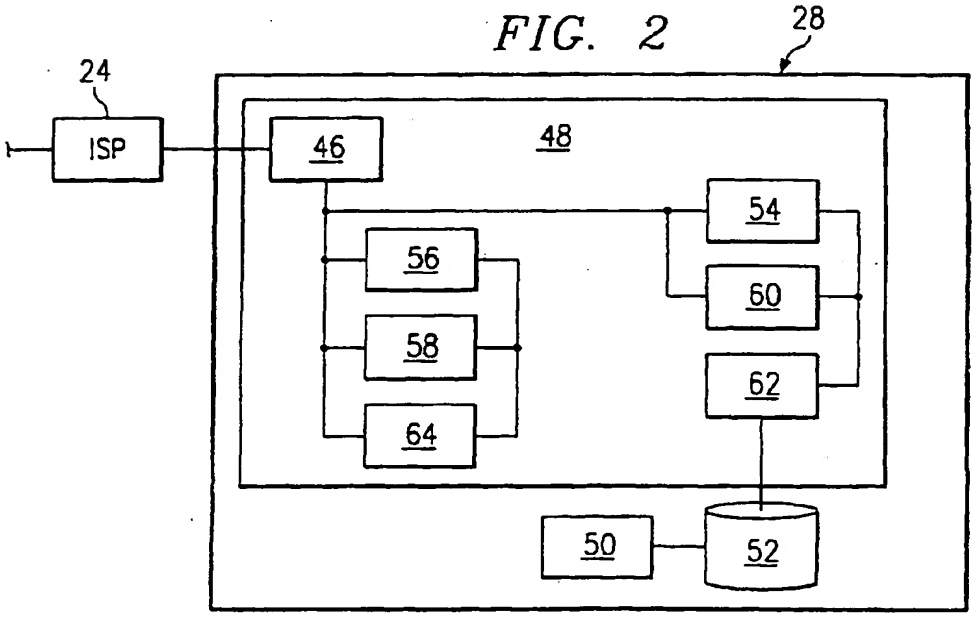


FIG. 2

Newly filed

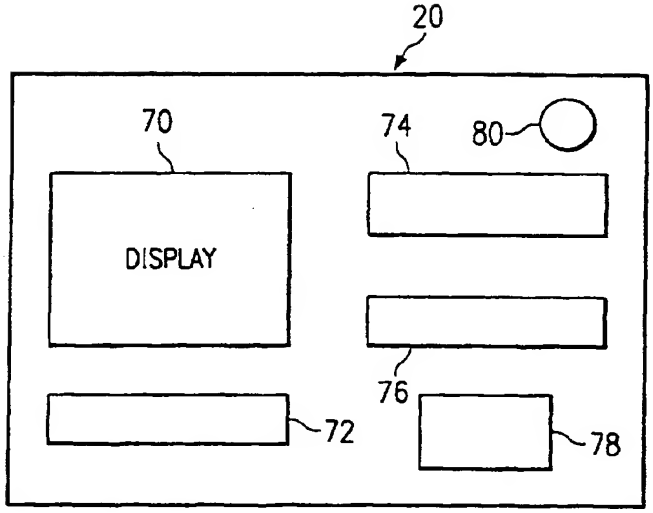


FIG. 3

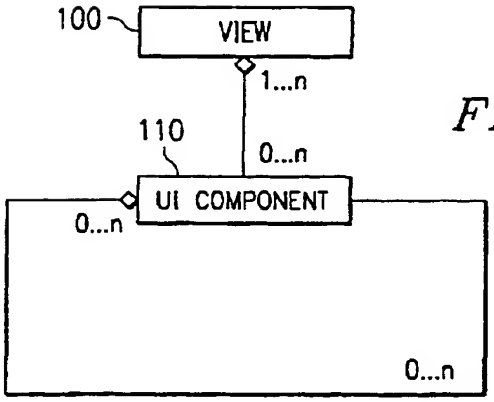


FIG. 4

FIG. 6

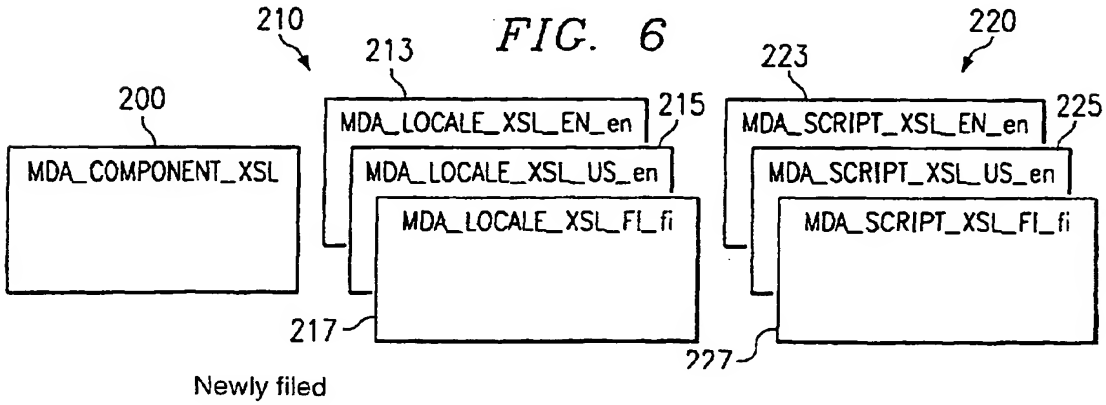
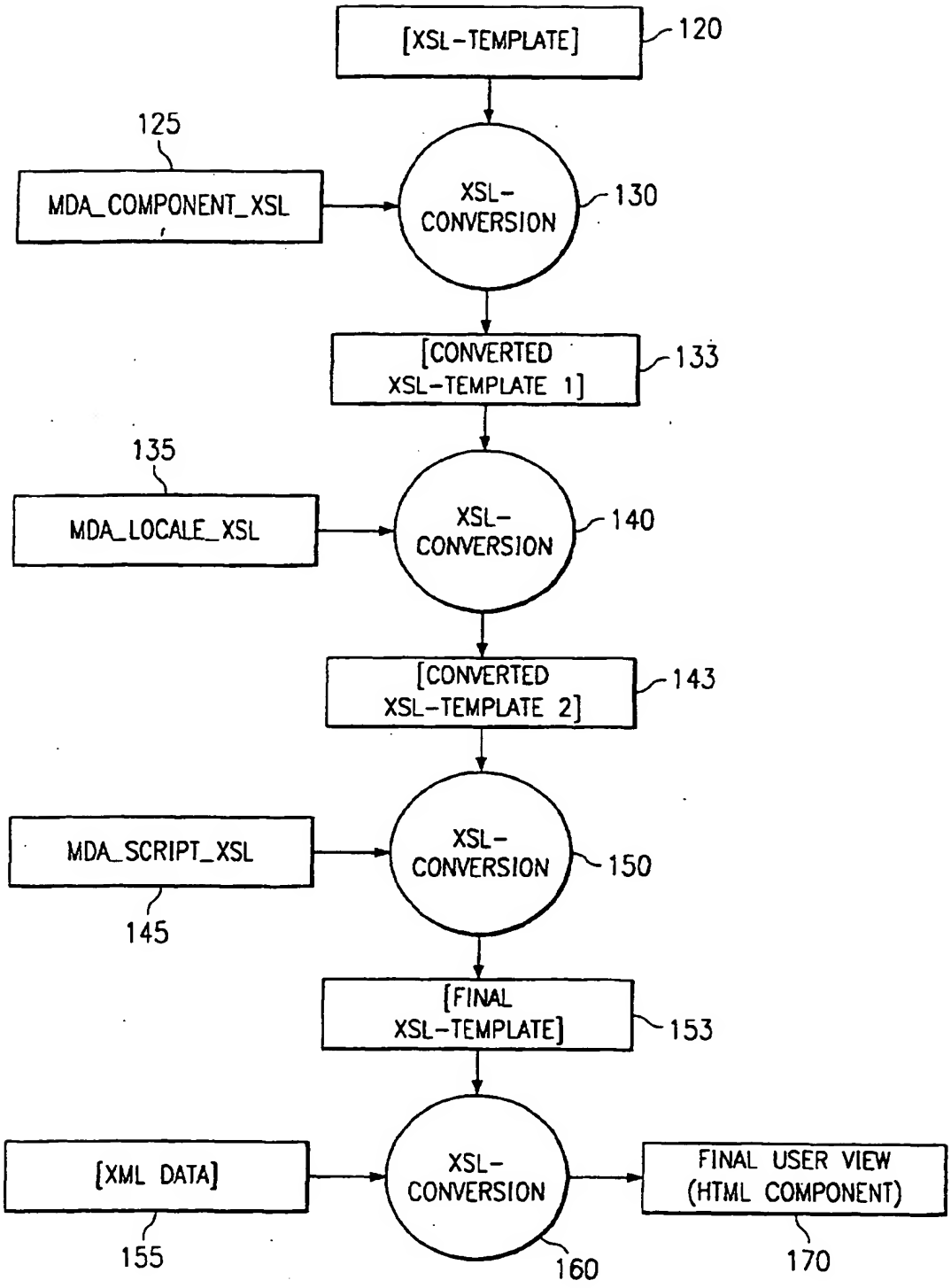


FIG. 5



Newly filed

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 168 162 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
06.11.2002 Bulletin 2002/45

(51) Int Cl.7: **G06F 9/44**

(43) Date of publication A2:
02.01.2002 Bulletin 2002/01

(21) Application number: **01304358.3**

(22) Date of filing: **16.05.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **30.06.2000 US 607369**

(71) Applicant: **Nokia Corporation**
02150 Espoo (FI)

(72) Inventor: **Paajanen, Samu**
00150 Helsinki (FI)

(74) Representative: **Read, Matthew Charles et al**
Venner Shipley & Co.
20 Little Britain
London EC1A 7DH (GB)

(54) Tag-based user interface

(57) The invention provides a method for converting one machine readable language into another by applying instructions to style templates. The invention provides a high level construct created from a set of lower lever tags. A man machine interface (MMI) may be created by using said constructs.

EP 1 168 162 A3

Printed by Jouve, 75001 PARIS (FR)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 30 4358

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	CAWSEY A: "Presenting tailored resource descriptions: will XSLT do the job?" COMPUTER NETWORKS, ELSEVIER SCIENCE PUBLISHERS B.V., AMSTERDAM, NL, vol. 33, no. 1-6, June 2000 (2000-06), pages 713-722, XP004304803 ISSN: 1389-1286 * page 714, right-hand column, last paragraph - page 716, right-hand column, paragraph 1; figures 1-7 * * page 719, left-hand column, line 1 - right-hand column, line 2 * * page 719, right-hand column, line 35 - line 39 * * page 720, left-hand column, line 12 - line 15 *	1-5,7-9	G06F9/44
X	"PARAMETERIZED XSL STYLE SHEETS" RESEARCH DISCLOSURE, KENNETH MASON PUBLICATIONS, HAMPSHIRE, GB, no. 423, July 1999 (1999-07), pages 1009-1011, XP000889028 ISSN: 0374-4353 * page 1009, line 22 - line 33 *	1-5,7-9	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F
A	BHAVEN SHAH: "Presenting XML to the Web" XML JOURNAL, SYS-CON MEDIA, ISSN: 1534-9780, vol. 1, no. 1, March 2000 (2000-03), pages 18-23, XP002211938 USA * page 19, left-hand column, line 1 - right-hand column, line 22; figure 2 *	1-9	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	2 September 2002	Fonderson, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)